The Inhibiting Bisection Problem*

Ali Pinar†
Computational Research Div.
Lawrence Berkeley Natl Lab.

Yonatan Fogel
Dept. Computer Science
SUNY

Bernard Lesieutre
Environmental Energy Tech. Div.
Lawrence Berkeley Natl. Lab

Abstract

Given a graph where each vertex is assigned a generation or consumption volume, we try to bisect the graph so that each part has a significant generation/consumption mismatch, and the cutsize of the bisection is small. Our motivation comes from the vulnerability analysis of distribution systems such as the electric power system. We show that the constrained version of the problem, where we place either the cutsize or the mismatch significance as a constraint and optimize the other, is NP-complete, and provide an integer programming formulation. We also propose an alternative relaxed formulation, which can trade-off between the two objectives and show that the alternative formulation of the problem can be solved in polynomial time by a maximum flow solver. Our experiments with benchmark electric power systems validate the effectiveness of our methods.

This paper is being submitted for a regular presentation.

*This work was supported by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of U.S. Department of Energy under contract DE-AC03-76SF00098.

†Contact author. One Cyclotron Road MS50F, Berkeley, CA 94720 Email: apinar@lbl.gov

1 Introduction

Robust operation of a distribution system requires identifying its vulnerabilities, that is likely events that can significantly reduce the transmission capability of the system. Many distribution systems such as the electric power, gas, and water distribution systems can be most naturally represented by graphs, where the generators/sources, consumers/sinks, and line/pipe intersections are represented by vertices, and lines and pipes are represented by edges. Most vulnerabilities in these systems stem from a loosely connected subsystem, with a significant generation/consumption mismatch. Due to this mismatch, this subsystem needs to be in strong interaction with the rest of the system. The lines or pipes providing this interaction are critical, since they need to operate at or close to their limits. The slack created by the failure of one of such these lines/pipes may not be picked up by the remaining lines, resulting in an overall system failure.

In this work, we address the problem of finding subgraphs with significant generation/load mismatch that can be separated from the rest of the graph by removing a small number of edges. Formally, we study the following problem.

> *Given a graph $G = (V, E)$ with a weight assignment on vertices (positive for generation and negative for consumption), find a decomposition of vertices into V_1 and $V_2 = V \setminus V_1$ so that the difference between the sum of weights of V_1 vertices and V_2 vertices is maximum and the number of edges between V_1 and V_2 vertices is minimum.*

We refer to this problem as the *inhibiting bisection* problem, since we are looking for the smallest bisection in the graph that maximally inhibits the generator to consumer distribution. The problem has two objectives, thus we can study two versions of the problem. In the *imbalance-constrained* version we look for a bisection with minimum number edges crossing between V_1 and V_2, while we make sure the difference between cumulative part weights is above a threshold. In the *cutsize-constrained* version, we maximize the difference between cumulative part weights, while the number of edges between the two parts is no more than a specified threshold. We prove that the inhibiting cut problem is NP-complete by reduction from the graph bisection problem. We also propose an integer programming formulation for this problem, which uses only $|V|$ binary variables. As an alternative to the two versions described above, we study a relaxed version that can trade-off between the two objectives through a user defined parameter that specifies the importance of one objective relative to the other. We propose a technique to solve this version of the inhibiting bisection problem in polynomial time by a maximum flow/minimum cut solver.

The inhibiting bisection problem is related to the graph bisection problem, but there is a fundamental difference: the graph bisection requires *balance* between the parts, whereas the inhibiting bisection problem requires *imbalance*. The balance constraint is what makes the graph bisection problem hard. If the sizes of the parts were irrelevant, the problem would be easier and polynomial time solvable as the minimum cut problem. In the inhibiting bisection problem, the sizes of the parts are relevant, but contrary to the graph bisection problem we are looking for an imbalanced decomposition. This difference hinders the applicability of tools designed for the graph bisection, often

referred to as graph partitioning, such as Chaco [8], Jostle [16], Metis [9], Party [14], and Scotch [11]. The local search strategies such "move" and "swap" techniques, or the multilevel paradigm are applicable, but not in a straightforward way.

Another related problem is the network inhibition problem, which is concerned with minimizing the transmission capability of the system with minimum effort [12]. While inhibition is the common theme, between the two problems, the inhibiting bisection problem looks for a bipartitioning of the of the vertices, whereas the network inhibition problem only tries to reduce the size of the minimum cut in the remaining graph.

Our work is mainly motivated by the vulnerability analysis of the electric power systems. The flow of power can be described by a system of nonlinear equations, and it is not possible to describe this flow merely in graph theoretical notation. Nevertheless, in our earlier studies, we showed that the Jacobian matrix that describes the feasibility boundary of power flow equations has the same structure as the Laplacian matrix in spectral graph theory [3, 4, 10]. This means that our model is accurate, when the equations are barely satisfied. Thus our graph model is effective for vulnerability analysis of the power system, even though, it cannot be used to accurately characterize the flow [13]. We applied our techniques to study a simplified model of the Western Interconnection system with 13,374 nodes and 16,520 power lines, and identified its vulnerabilities. Our techniques can solve an instance of the problem in less than a second, making it practically applicable.

The remainder of this paper is organized as follows. In Section 2, we define the inhibiting bisection problem, prove it is NP-complete, and provide an integer programming formulation for it. Section 3 studies a relaxed version of the problem, where we allow a trade off between the imbalance and the cutsize of the bipartitioning, and shows how this problem can be solved using a maximum flow solver. We present our experimental results in Section 4 and conclude with Section 5.

2 Preliminaries

In this section, we first define the notation used in the paper and introduce the inhibiting bisection problem. Then we prove the problem is NP-complete, and provide an integer programming formulation of the problem.

2.1 Problem Definition

Intuitively, in the *inhibiting bisection* problem, we look for the easiest way to isolate a significant portion of the generation from the consumption vertices. In other words, we look for a cut in the graph, which maximally inhibits its transmission capability. Given a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, a *cut* is defined by a bipartitioning of the vertices into V_1 and $V_2 = V \setminus V_1$. For a given partitioning of the vertices into V_1 and V_2, we define the *cutset* $C(V_1, V_2)$ as the set of edges that cross between the two parts. i.e.,

$$C(V_1, V_2) = \{(v_i, v_j) : (v_i, v_j) \in E \text{ and } v_i \in V_1 \text{ and } v_j \in V_2 \}$$

The *cutsize* refers to the number of edges in the cutset $|C(V_1, V_2)|$.

Each vertex $v_i \in V$ is associated with a weight w_i. When $w_i \geq 0$, the associated vertex v_i is a generation vertex, and it is a consumption vertex otherwise. We use $W(V)$ to refer to the cumulative weight of a set of vertices, i.e.,

$$W(V_1) = \sum_{v_i \in V_1} w_i$$

For simplicity, we assume the total generation is equal to the total consumption, i.e.,

$$W(V) = \sum_{v_i \in V} w_i = 0.$$

Definition 2.1 Inhibiting Bisection Problem.
Given a graph $G = (V, E)$, with weights on its vertices, and a bound on the cutsize B. Find a bipartitioning of V into V_1 and V_2 that maximizes $|W(V_1) - W(V_2)|$, while $|C(V_1, V_2)| \leq B$.

We refer to the version of the inhibiting cut problem defined above as the *cutsize-constrained* version, since we constrain the cutsize and seek for maximum imbalance between the two parts. We can define the *imbalance-constrained* version of the problem by switching the roles of the cutsize and imbalance, where we look for a cut with of minimum size, whose weight mismatch is no smaller than a specified threshold.

The traditional minimum cut problem is concerned with finding a cut that blocks all paths between a source and a terminal vertex. Whereas the inhibiting bisection problem allows some flow to the terminal to find cuts of smaller size. There are two other problems that are related to the inhibiting cut problem. The network inhibition problem aims at finding the most cost-effective way to attack a network to minimize it is transmission capability. In [12], Phillips defines the network inhibition problem as follows. Each edge in the network has a destruction cost, and a fixed budget is given to attack the network. A feasible attack removes a subset of the edges, whose total destruction cost is no greater than the budget, and the network inhibition problem is to find an attack that minimizes the value of a maximum flow in the graph after the attack. While the network inhibition problem and the inhibiting cut problems are similar in flavor, there is a fundamental difference between the two, since the inhibiting cut problem looks for a bipartitioning of the vertices, whereas the network inhibition problem only tries to reduce the size of the minimum cut in the residual graph. Hence, a subset of the cutset of a solution to the inhibiting cut problem can be a solution to the network inhibition problem. The network inhibition problem is NP-Complete [12]. Phillips provides a comprehensive study on network inhibition problem [12].

The graph bisection problem also seeks for a bipartitioning of vertices with minimum cutsize, however the constraint there is that the cumulative weights of the two parts should be as even as possible. Graph bisection problem is well-studied, and is known to be NP-complete [7]. Numerous algorithms for this problem have yielded effective software tools, however these techniques are based on finding a balanced cut, therefore the algorithmic techniques and the associated software tools are only of limited applicability for our problem.

2.2 Complexity

In this section, we prove that the inhibiting cut problem is NP-complete by reduction from the graph bisection problem. The graph bisection problem is defined in [6] as follows.

Definition 2.2 Graph Bisection Problem.
Given a graph $G = (V, E)$ with two designated vertices s and t, and integer $B \leq |E|$, is there a partition of the nodes into two sets S and $T = V \setminus S$ such that $s \in S$, $t \in T$, $|S| = |T| = |V|/2$ (or within 1 for $|V|$ odd), and the number of edges with one endpoint in set S and the other endpoint in set T is at most B?

Garey, Johnson, and Stockmeyer proved this problem is NP-complete [7]. Note that the definition above of the graph bisection does not include vertex weights, but the NP-completeness of the weighted version is implied, since the unweighed version is only an instance of the weighted version.

Theorem 2.1 *The inhibiting bisection problem is NP-complete.*

Proof: First we pose the inhibiting bisection problem as a decision problem as follows.

Given a graph $G = (V, E)$, a weight w_i for each vertex $v_i \in V$ and bounds B_w and B_c. Decide if there exists a partitioning of V into V_1 and V_2 so that $|W(V_1) - W(V_2)| \geq B_w$ and $C(V_1, V_2) \leq B_c$.

It is easy to check the correctness of a given solution in polynomial time, thus the problem is in NP. We show how to solve an instance of the graph bisection problem using an algorithm for the inhibiting bisection problem to prove NP-completeness. Our construction enforces that the vertices of the original graph are bisected evenly by using the imbalance constraint. And we define the new cutsize threshold so that the cutsize constraint of the inhibiting bisection problem is satisfied only if the cutsize of the bisection of the original graph is below its threshold B. For simplicity presentation, we assume $|V|$ is even, since the proof can be easily generalized for the odd case.

Given a graph $G = (V, E)$ and edge cut bound B as an instance of the graph bisection problem, we define the graph $G' = (V', E')$ for the inhibiting bisection problem as follows. The vertex set V' includes all vertices of the original graph, and we assign a weight of 1 to each. Then, we add a set of $|V|^2$ auxiliary vertices, A, each with weight $M < -|V|$, and another vertex, t, with weight $M|V|^2 - |V|$. Observe that the maximum imbalance is achieved when all auxiliary vertices are in one part and vertex t and all vertices of the original graph are on the other. Formally, $V' = V \cup A \cup \{t\}$, and weight w_i of vertex v_i is defined as

$$w_i = \begin{cases} -M & \text{if } v_i \in A \\ M|V|^2 - |V| & \text{if } v_i = t \\ 1 & \text{if } v_i \in V \end{cases} .$$

To enforce a balanced distribution of the vertices of the original graph, we add edges from each auxiliary vertex in A to each vertex of the original graph in V. Formally,

$$E' = E \cup \{(a_i, v_i) : \ a_i \in A, v_i \in V\}$$

Observe that the cutsize is minimum when all vertices of the original graph are in the same part as A vertices. However, the imbalance is maximized when vertices of the original graph are in the opposite part with the A vertices. We set the imbalance threshold B_w for the inhibiting bisection problem so that all vertices in A are in part V_1 and t is in part V_2, and we use the cutsize threshold B_c to ensure that vertices of the original graph are evenly bisected between the two parts.

Formally, we claim that there is a bisection of the graph $G = (V, E)$ with cutsize no larger than B, if and only if there is a bisection of the graph $G' = (V', E')$ with cutsize $B_c \leq |V|^3/2 + B$, and imbalance $B_w \geq 2M|V|^2 - |V|$.

Without loss of generality, let $t \in V_2$. First observe that imbalance constraint is only satisfied when vertex t and A vertices are assigned to different parts. This brings an imbalance of $2M|V|^2-|V|$. Each vertex of the original graph assigned to V_1 decreases this imbalance by 1, and each assigned to V_2 increase this imbalance by 1. Thus, for the constraint on B_w to be satisfied, at least $|V|/2$ vertices must be assigned to V_2. On the other hand, each vertex of the original graph assigned to V_2 contributes at least $|V|^2$ edges to the cutset, since it is connected to all A vertices. Thus, the constraint B_c requires at least $|V|/2$ vertices to be on V_1. Therefore, the constraints on B_c and B_w are both satisfied only when V_1 and V_2 both contain exactly $|V|/2$ vertices, which guarantees that the balance constraint of the graph bisection problem is satisfied. The even distribution of vertices contributes $|V|^3/2$ edges to the cutset, regardless of the particular bisection, which leaves a limit of B for edges between the vertices of the original graph. Thus the cutsize constraint for the graph bisection problem is satisfied, only if the cutsize constraint for the inhibiting bisection problem is satisfied. ∎

2.3 An Integer Programming Formulation

Let $G = (V, E)$ be a graph, and let A be the $|E| \times |V|$ node-arc incidence matrix of this graph, where each row of A represents an edge of the graph, and each column represents a vertex. The matrix A has two nonzeros in each column: a 1 at one of the vertices of the respective edge. and a -1 on its other vertex. We define a binary variable p_i for each vertex $v_i \in V$, so that

$$p_i = \begin{cases} 1 & \text{if } v_i \in V_1 \\ 0 & \text{if } v_i \in V_2 \end{cases},$$

where V_1 and $V_2 = V \setminus V_1$ denote the partitioning of V that defines the cut. We also define a binary variable c_i for each edge, so that

$$c_i = \begin{cases} 1 & \text{if } e_i \in C(V_1, V_2) \\ 0 & \text{otherwise} \end{cases}$$

The imbalance $|W(V_1) - W(V_2)|$ can be determined by only looking at one of $W(V_1)$ and $W(V_2)$, since the weights sum up to 0, and thus $W(V_1) = -W(V_2)$. Without loss of generality, we assume $W(V_1) > 0$. The inhibiting bisection problem can then be formulated as follows.

$$\max_{p,c} \quad p^T w \tag{1}$$

$$s.t. \quad Ap - c \leq 0 \tag{2}$$

$$Ap + c \geq 0 \tag{3}$$

$$e^T c \leq B \tag{4}$$

$$p_i \in \{0,1\} \quad \text{for } i = 1, 2, \ldots, |V| \tag{5}$$

$$c_i \in \{0,1\} \quad \text{for } i = 1, 2, \ldots, |E| \tag{6}$$

Here, the objective function, $W(V_1) = p^T w = \sum_i^{|V|} p_i w_i$ maximizes the imbalance. Constraints (5) and (6) guarantee that the p and c are binary variables. In constraint (4), e refers to a vector of all 1s, and thus $e^t c = \sum_{i=1}^{|E|}$ is equal to the cutsize, guaranteeing that the the cutsize is no more than the specified limit. Constraints (2) and (3) are used to enforce any edge that connects the two parts to be labeled as a cut edge. Consider an edge e_k that goes from v_i to v_j, for which we will have the two constraints

$$p_j - p_i - c_k \leq 0 \tag{7}$$

$$p_j - p_i + c_k \geq 0 \tag{8}$$

We need to show that $c_k = 1$, if v_i and v_j are on different parts ($p_i \neq p_j$). If $p_i = 1$ and $p_j = 0$, (7) forces c_k to be ≥ 1. Symmetrically, If $p_i = 0$ and $p_j = 1$, it will be (8) that forces c_k to be ≥ 1. Thus, c_k is marked with 1 if its end points are on different parts.

When the two vertices are in the same part, i.e., $p_i = p_j$, c_k can be either zero or one. However, since we try to limit the number of cut edges with (4), c_k will be 1 only if removing an extra edge does not increase the imbalance. Also note that the solution will still be feasible, even when an edge is falsely marked as a cut edge.

This analysis further shows that, we do not need to impose c variables to be binary explicitly. When the edge e_k is on the cut, we need $c_k \geq 1$, and when e_k is an internal edge, c_k will be at its minimum due to constraint (4). In an optimal solution to (1)–(6), c variables will naturally take binary values, when they are constrained to be in the $[0, 1]$ region. Therefore we can replace (6) with

$$0 \leq c_i \leq 1 \quad \text{for } i = 1, 2, \ldots, |E|$$

This improved formulation reduces the number of binary variables to $|V|$, whereas the initial formulation employed $|V| + |E|$ binary variables.

The formulation in (1)–(6) corresponds to the cutsize constrained version of the inhibiting bisection problem. For the imbalance constrained version of the problem we can make the $e^T c$ the objective function to minimize, and add another constraint as $p^t w > B$.

3 Trade-off between Imbalance and Cutsize

The inhibiting bisection problem is naturally a dual objective problem. We need a bisection of minimum cutsize with largest imbalance. While dealing with such dual objective problems, the common practice is to move one of the objectives into the constraints, and look for an optimal solution for the other, as we did in the previous section. An alternative way is to define a relative importance of one objective with respect to the other and define a single objective function. For example we can redefine the objective of the inhibiting bisection problem as (ϵ)cutsize- $(1-\epsilon)$imbalance, where $0 \leq \epsilon \leq 1$ represents the relative importance of cutsize compared to the imbalance (notice that we negate the imbalance to have a minimization problem). The disadvantage of this approach lies in the difficulty of the definition of ϵ. While it is hard to come up with an exact number for a dual objective problem, this approach is very useful when the solution method is fast enough so that it enables analysis of a whole spectrum for ϵ.

In this section, we study this relaxed version of the inhibiting bisection problem, where we work with a single objective function that includes both the imbalance and the cutsize of a bisection. Surprisingly, this version of the inhibiting bisection problem can be solved in polynomial time, even though the cutsize and imbalance-constrained versions of the problem are NP-complete. Our solution to the dual-objective version of the inhibiting bisection problem uses a reduction to the maximum flow/ minimum cut problem. Next we review the minimum cut problem, and then describe how the inhibiting bisection problem can be solved using a maximum flow/minimum cut solver.

3.1 Maximum Flow/ Minimum Cut Problem

A flow network $G = (V, E)$ is defined a set of vertices V, a set of edges E, where each edge (u, v) has a nonnegative capacity $c(u, v)$, and two special vertices: a *source* s and a *terminal* t. A *flow* in G is a real valued function, $f : E \rightarrow \mathcal{R}$. We use $f(u, v)$ to refer to a flow on the edge from vertex u to vertex v. A single source vertex, s, is used, which is connected to all other vertices with production, and the capacity of the connecting edge is equal to the production on that node. Similarly, only a single terminal vertex, t, is used, which is connected to all other vertices with consumption, and the capacity of the connecting edge is equal to the consumption on that node. We say a flow is feasible if it respects conservation of flow and the capacity constraints on edges. Conservation of flow requires that the total flow into a node is equal to total flow out of that node is equal. The value of a flow is defined by the total flow leaving the source, and the maximum flow problem tries to find a feasible flow with maximum value.

A closely related concept to maximum flow is the *minimum cut* in a graph. A *cut* in a graph is defined by a bipartitioning of vertices V into S and $T = V \setminus S$, so that $s \in S$ and $t \in T$. We say an edge is *on the cut* is one of its end vertices is in S and the other is in T. The *capacity* of a cut is defined as the sum of capacities of the edges on the cut, and a minimum cut is the one with minimum capacity among all the cuts.

It is easy to see that the capacity of any cut is an upper bound the value of a maximum flow, since

the edges on the cut block all paths from the source to the terminal, and thus the total flow cannot exceed their capacity. Clearly, the capacity of the minimum cut will provide the tightest bound on the value of maximum flow. As one of the earliest and fundamental results in combinatorial algorithms, Ford and Fulkerson proved that the capacity of a minimum cut is actually equal to the value of a maximum flow. This duality between maximum flow and minimum cut underlies many algorithms for flow problems. A more detailed discussion on network flow algorithms can be found in [2, 15].

3.2 Reducing the Inhibiting Bisection Problem to the Minimum Cut Problem

In this section, we show the relaxed version of the inhibiting bisection problem can be solved using a maximum flow solver. Specifically, we try to solve the following problem.

> *Given a graph $G = (V, E)$, with weights on its vertices, and a trade-off value $0 \leq \epsilon \leq 1$. Find a bipartitioning of V into V_1 and V_2 that minimizes*
>
> $$\epsilon|C(V_1, V_2)| - (1 - \epsilon)|W(V_1) - W(V_2)| \tag{9}$$

Note that the imbalance entry is placed as a negative entry, since we have a minimization problem.

Given an instance of the inhibiting bisection problem with $G = (V, E)$ and ϵ, we define a flow graph by adding a source vertex s and a terminal t to the vertex set. We connect each generating vertex v_i ($w_i \geq 0$) to the source vertex s and each consuming vertex v_i ($w_i < 0$) to the terminal t. These new edges will help us encode the imbalance information in the edge capacities, so that a minimum capacity cut in the new graph minimizes (9).

We define the capacity of each edge as the change in (9) when this edge is on the cutset of the bisection. The imbalance is maximized when all generating vertices are on one part, and all consuming vertices are in the other. In our construction, the edges between the source vertex and the generating vertices pull the generating vertices to S in the minimum-cut bisection, and similarly, edges between the terminal vertex t and the consuming vertices pull the consuming vertices to T. Therefore, after the bisection, S corresponds to the generation-rich side, while T corresponds to the load-rich side. An edge between the source vertex and a generating vertex v_i being cut means this generating vertex is being placed on the load-rich side, which decreases the imbalance number by $2w_i$, and the objective function by $(1-\epsilon)2w_i$, which we assign as the capacity of this edge. Similarly, an edge between the terminal vertex and a consuming vertex v_i being cut means this consuming vertex is being placed on the generation-rich side, which decreases the imbalance number by $2w_i$, and the objective function by $(1 - \epsilon)2w_i$, which we assign as the capacity of this edge. If an edge inherited from the original graph is cut, only the first term in (9) is affected, thus we assign ϵ as the capacity of each edge from the original graph.

A maximum flow solver can find a minimum capacity cut in this graph, which by definition minimizes (9).

Figure 1: Inhibiting bisection analysis of the Western states grid with integer programming

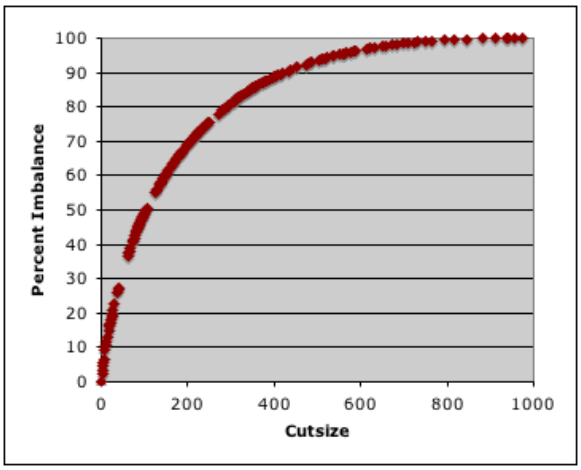

Figure 2: Inhibiting bisection analysis of the Western states grid with the relaxed formulation

4 Experimental Results

To validate the effectiveness of our proposed techniques, we have applied them on a benchmark power system: a simplified model of Western states power grid, which has 13374 nodes and 16,520 power lines. We have applied the integer programming formulation in Section 2.3, and the relaxed formulation, which can be solved with a maximum solver as described in Section 3.2. We have used PICO (Parallel Integer and Combinatorial Optimization) [5], developed at Sandia National Laboratories to solve the integer programming problems. As the maximum flow solver, we used the implementation of the push relabel algorithm by Cherkassky and Goldberg [1].

The results are presented in Figure 1 and 2. In these figures, the x-axes correspond to the cutsize and y-axes correspond to the percent imbalance, which is computed as

$$\frac{|W(V_1)| * 100}{W_T}$$

where W_T represent the total production in the system, i.e., $W_T = (\sum_i |w_i|)/2$. Figure 1 presents

results for varying values of teh cutsize bound, whereas Figure 2 presents 1000 experiments with ϵ taking equal length steps from 0 to 1.

As expected, the imbalance of the cuts increase sharply with increasing cutsize at first (lower-left portion), then starts to level off (middle portion), and then stays the same, when all generators are isolated from the rest of the graph. The lower left portion of the figure shows system vulnerabilities that can happen and cause a severe disturbance. Some of these cuts are trivial, and some are already known by those who know the system while some others were surprises even for the system experts. While the cuts in the middle of the figure are not as likely to happen, they provide valuable information for finding fundamental weaknesses of the system. We observed that the cuts in the middle of the figure are composed of combinations of smaller cuts, which can help us identify critical load corridors in the system. These corridors may not be critical for the system at present, but the system will be vulnerable if these corridors are not taken into account while the active generators and their generation levels are being decided. The two formulations often identify the same vulnerabilities, but the integer programming formulation is easier from a user's perspective it is easier to specify the problem.

We have performed our experiments with the minimum cut formulation on a computer with a 3.2 Mhz Xeons processor with 2MB cache. And each run of the minimum cut algorithm took only 0.3 seconds on average on the system with 13,374 nodes, and 16,520 edges. The IP solvers took much more time as expected, but we are providing a detailed performance analysis here, since neither the code was well-tuned, nor such a comparison is really necessary.

5 Conclusions

Many distribution systems such as the electric power, gas and water systems can be most naturally represented as graphs. This enables adoption of graph theoretical techniques for the vulnerability analysis of these systems. We introduced the inhibiting bisection problem, motivated by such a vulnerability analysis. The inhibiting bisection problem seeks a subgraph with a significant generation/consumption mismatch, which is connected to the rest of the graph with a small number of edges. We show that the constrained version of the problem, where we place either the cutsize or the mismatch significance as a constraint and optimize the other, is NP-complete, and provide an integer programming formulation. We also propose an alternative formulation, which can trade-off between the two objectives, and show that the alternative formulation of the problem can be solved with a maximum flow solver. Our experiments with benchmark electric power systems validate the effectiveness of our methods.

This paper reports our initial results, and leaves some areas for future work. First we plan to study the relation between the two formulations better. In particular, the relaxed version can be used as an ϵ approximation algorithm for the constrained version. What is more intriguing is the relation between inhibiting bisection problem and the network inhibition problem. We are planning to study how these problems relate to each other. Finally, we are planning to apply our techniques to different graphs, especially those representing different distribution systems.

Acknowledgments

The authors would like to thank Vaibhav Donde, from ABB Inc., Juan Meza, Chao Yang from Lawrence Berkeley Laboratory, Sandip Roy from Washington State University, Adam Reichert from University of Illinois for many helpful discussions.

References

[1] Boris V. Cherkassky and Andrew V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[3] Vaibhav Donde, Vanessa Lopez, Bernard Lesieutre, Ali Pinar, Chao Yang, and Juan Meza. Identification of severe multiple contingencies in electric power systems. *IEEE Transactions on Power Systems*. submitted.

[4] Vaibhav Donde, Vanessa Lopez, Bernard Lesieutre, Ali Pinar, Chao Yang, and Juan Meza. Identification of severe multiple contingencies in electric power networks. In *Proceedings of the 37th North American Power Symposium*, Ames, Iowa, 2005.

[5] J. Eckstein, W. Hart, and C. Phillips. Pico: An object-oriented framework for parallel branch-and-bound, inherently parallel algorithms in feasibility and optimization and their applications. *Elsevier Scientific Series on Studies in Computational Mathematic s*, pages 219–265, 2001.

[6] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 2000.

[7] Michael R. Garey, David S. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[8] Bruce Hendrickson and Robert Leland. The chaco user's guide: Version 2.0. Technical Report SAND94-2692, Sandia National Laboratories, 1994.

[9] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. In *Proc. International Conference on Parallel Processing*, pages 113–122, 1995.

[10] Bernard Lesieutre, Sandip Roy, Vaibhav Donde, and Ali Pinar. Power sytem extreme event screening using graph partitioning. In *Proceedings of the 38th North American Power Symposium*, Carbondale, Illinois, 2006.

[11] F. Pellegrini. Scotch 4.0 users guide. Technical report, Research report Laboratoire Bordelais de Recherche en Informatique, 2006.

[12] Cynthia A. Phillips. The network inhibition problem. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 776–785, New York, NY, USA, 1993. ACM Press.

[13] Ali Pinar, Vaibhav Donde, Bernard Lesieutre, Juan Meza, and Yonatan Fogel. Vulnerability analysis of the electric power grid. *Siam Optimization*. to be submitted.

[14] R. Preis and R. Diekmann. The party partitioning-library, user guide - version 1.1. Technical Report tr-rsfb-96-024, University of Paderborn, 1996.

[15] Robert E. Tarjan. *Data Structures and Network Algorithms*. Siam Press, 1998.

[16] C. Walshaw, M. Cross, and K. McManus. Multiphase mesh partitioning. *Appl. Math. Modelling*, 25:123–140, 2000.